
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la programación 02</i>				
TÍTULO DE LA PRÁCTICA:	<i>HashMap</i>				
NÚMERO DE PRÁCTICA:	<i>08</i>	AÑO LECTIVO:	<i>2024-B</i>	NRO. SEMESTRE:	<i>//</i>
FECHA DE PRESENTACIÓN	<i>29/11/2024</i>	HORA DE PRESENTACIÓN	<i>18:00:00</i>		
INTEGRANTE (s) <i>Mauro Snayder Sullca Mamani</i>				NOTA (0-20)	
DOCENTE(s): <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS	
<p>I. EJERCICIO RESUELTO:</p> <pre> 11 public class Soldado { 12 // Creamos los atributos 13 private String nombre= " "; 14 private int vida; 15 private int fila; 16 private int columna; 17 private String color; 18 19 // Creamos los Set y Get de cada atributo 20 public String getNombre() { 21 return nombre; 22 } 23 24 public void setNombre(String nombre) { 25 this.nombre = nombre; 26 } 27 28 //Metodo para saber a que tipo de ejercito pertenece el soldado 29 public void setColor(String color){ 30 this.color=color; 31 } 32 33 public int getVida() { 34 return vida; 35 } 36 37 //Metodo para darle color a "vida" 38 public void printVida(){ 39 System.out.print(color+vida+"\u001B[0m"); 40 } </pre>	

```

41
42 public int getFila(){
43     return fila;
44 }
45
46 public int getColumna(){
47     return columna;
48 }
49 //Generamos una posicion aleatoria para el soldado
50 public void aleatorioPosicion(int fila,int columna){
51     this.fila=(int) (Math.random()*fila);
52     this.columna=(int) (Math.random()*columna);
53 }
54 //Generamos la vida del soldado
55 public void aleatorioVida(){
56     this.vida=(int) (Math.random()*5+1);
57 }
58 // Creamos el toString
59 public String toString() {
60     return "Soldado(" + "nombre=" + nombre + ", vida=" + vida + ", fila=" + fila + ", columna=" + columna + ")\n";
61 }
62 }

```

```



9  * @author Mauro Snyder
10  */
11  import java.util.*;
12
13  public class VideoJuego5 {
14      public static void main(String[] args) {
15          Scanner scan=new Scanner(System.in);
16          while(true){//Creamos un bucle para hacerlo iterativo
17              Soldado[][] tabla=new Soldado[10][10];//Creamos la tabla
18              HashMap<String,Soldado> ejercito1=new HashMap<String,Soldado>();//Creamos HashMap para ejercito 1
19              HashMap<String,Soldado> ejercito2=new HashMap<String,Soldado>();//Creamos HashMap para ejercito 2
20              for (int i=0;i<tabla.length;i++){//Inicializamos el tablero con valores vacios
21                  for (int j=0;j<tabla[i].length;j++){
22                      tabla[i][j]=new Soldado();
23                  }
24              }
25              inicializarEjercito(tabla,ejercito1,"\033[1;31m");//El ejercito 1 será de color rojo
26              inicializarEjercito(tabla,ejercito2,"\033[1;34m");//El ejercito 2 sea de color azul
27              mostrarEjercitoTabla(tabla);
28              System.out.print("\nEl soldado con mayor vida del ejercito 1 es: "+mayorVida(ejercito1).toString());
29              System.out.print("\nEl soldado con mayor vida del ejercito 2 es: "+mayorVida(ejercito2).toString());
30              System.out.println("\nEl promedio de la vida del ejercito 1 es: "+vidaPromedio(ejercito1));
31              System.out.println("\nEl promedio de la vida del ejercito 2 es: "+vidaPromedio(ejercito2));
32              System.out.println("\nLista de los soldados por orden de creacion: ");
33              mostrarEjercitoOrdenCreacion(ejercito1,1);
34              mostrarEjercitoOrdenCreacion(ejercito2,2);
35              System.out.println("\nEl ranking de los soldados es: ");
36              rankingSoldadosV1(ejercito1,1);
37              rankingSoldadosV2(ejercito2,2);
38              ganador(ejercito1,ejercito2);
39              System.out.println("\nDesea generar otros ejercitos(?) 1=Si 0=No");
40              int opcion=scan.nextInt();
41              if (opcion==0)
42                  break;
43          }
44      }
45      // Método para inicializar una tabla con cierto numeros de soldados
46      public static void inicializarEjercito(Soldado[][] tabla,HashMap<String,Soldado> ejercito,String color){
47          int numSoldados=(int) (Math.random()*10+1);//Generamos la cantidad de soldados
48          for (int i=0;i<numSoldados;i++){
49              Soldado persona=new Soldado();//creamos "persona" para despues ponerlo dentro del tablero y del HashMap "ejercito"
50              persona.setColor(color);//ponemos el color al soldado "persona"
51              do {
52                  persona.aleatorioPosicion(tabla.length,tabla[0].length);//generamos una posicion al soldado "persona"
53                  persona.aleatorioVida();//generamos la vida al soldado "persona"
54                  persona.setNombre("soldado"+ " "+persona.getFila()+"X"+persona.getColumna());//Generamos el nombre del soldado "persona"
55              }

```

```

56 //El bucle se repite si en una posicion aleatoria ya existe un soldado puesto
57 while(!tabla[persona.getFila()][persona.getColumna()].getNombre().equals(""));
58 ejercito.put(persona.getNombre(), persona); //ponemos el soldado "persona" dentro del HashMap "ejercito"
59 tabla[persona.getFila()][persona.getColumna()] = persona; //ponemos el soldado dentro del tablero
60 }
61 }
62 // Metodo para mostrar la tabla
63 public static void mostrarEjercitoTabla(Soldado[][] tabla) {
64     System.out.println("
65     for (int i=0; i<tabla.length; i++) {
66         System.out.print("
67         for (int j=0; j<tabla[i].length; j++) {
68             if (tabla[i][j].getVida()==0) //Si la vida es 0, entonces no hay un soldado en esa posicion
69                 System.out.print("
70             else { //Caso contrario, si existe un soldado en esa posicion
71                 System.out.print("
72                 tabla[i][j].printVida(); //imprimimos la vida del soldado con su color correspondiente
73                 System.out.print("
74             }
75         }
76         System.out.println();
77         System.out.println("
78     }
79 }
80 //Metodo para determinar el soldado con mayor vida
81 public static Soldado mayorVida(HashMap<String, Soldado> ejercito) {
82     Soldado mayor = new Soldado(); //creamos un objeto para almacenar al soldado mayor
83     for (Soldado persona: ejercito.values()) { //Recorremos todos los valores del HashMap "ejercito"
84         if (persona.getVida() > mayor.getVida()) //Buscamos al soldado con mayor vida
85             mayor = persona;
86     }
87     return mayor;
88 }
89 //Metodo para determinar la vida total de todos los soldados
90 public static double vidaPromedio(HashMap<String, Soldado> ejercito) {
91     double vidaT = 0; //vida inicial
92     for (Soldado persona: ejercito.values()) { //Recorremos todos los valores del HashMap "ejercito"
93         vidaT += persona.getVida(); //Sumamos las vidas de todos los soldados
94     }
95     return vidaT / ejercito.size(); //retornamos la suma total de vidas, dividido por la cantidad de soldados (promedio)
96 }
97 //Metodo para ver las lista de los soldados por el orden de creacion
98 public static void mostrarEjercitoOrdenCreacion(HashMap<String, Soldado> ejercito, int tipo) {
99     System.out.println("Ejercito "+tipo+" : ");
100     for (Soldado persona: ejercito.values()) { //imprimimos todos los soldados del HashMap "ejercito"
101         System.out.print(persona.toString());
102     }
103 }
104 //Metodo para ver el ranking de los soldados version 1 (por vida)
105 public static void rankingSoldadosV1(HashMap<String, Soldado> ejercito, int tipo) {
106     //ordenamos el HashMap y almacenamos los valores en un ArrayList
107     ArrayList<Soldado> ejercitoList = ordenamientoBurbuja(ejercito);
108     System.out.println("Ejercito "+tipo+" : ");
109     for (int i = ejercitoList.size()-1; i >= 0; i--) { //imprimimos el ArrayList
110         System.out.print(ejercitoList.get(i).toString());
111     }
112 }
113 //Metodo para ver el ranking de los soldados version 2 (por vida)
114 public static void rankingSoldadosV2(HashMap<String, Soldado> ejercito, int tipo) {
115     //ordenamos el HashMap y almacenamos los valores en un ArrayList
116     ArrayList<Soldado> ejercitoList = ordenamientoInsercion(ejercito);
117     System.out.println("Ejercito "+tipo+" : ");
118     for (int i = ejercitoList.size()-1; i >= 0; i--) { //imprimimos el ArrayList
119         System.out.print(ejercitoList.get(i).toString());
120     }
121 }
122 //Metodo de ordenamiento Burbuja para la vida de los soldados
123 public static ArrayList<Soldado> ordenamientoBurbuja(HashMap<String, Soldado> lista) {
124     ArrayList<Soldado> list = new ArrayList<Soldado>(lista.values()); //Copiamos los valores de HashMap en un ArrayList
125     Soldado cambio;
126     //Luego hacemos el respectivo ordenamiento
127     for (int i = 0; i < list.size()-1; i++) {
128         for (int j = 0; j < list.size()-i-1; j++) {
129             if (list.get(j).getVida() > list.get(j+1).getVida()) {
130                 cambio = list.get(j);
131                 list.set(j, list.get(j+1));
132                 list.set(j+1, cambio);
133             }

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

```

134         }
135     }
136     return list;
137 }
138 //Metodo de ordenamiento de insercion para la vida de los soldados
139 public static ArrayList<Soldado> ordenamientoInsercion(HashMap<String,Soldado> lista) {
140     ArrayList<Soldado> list=new ArrayList<Soldado>(lista.values()); //Copiamos los valores de HashMap en un ArrayList
141     //Luego hacemos el respectivo ordenamiento
142     for (int i=1;i<list.size();i++){
143         Soldado soldadoActual=list.get(i);
144         int j=i-1;
145         while (j>=0 && list.get(j).getVida(>soldadoActual.getVida()){
146             list.set(j+1,list.get(j));
147             j--;
148         }
149         list.set(j+1,soldadoActual);
150     }
151     return list;
152 }
153 //Determinar el ganador de la batalla (por la cantidad de soldados que tiene cada ejercito)
154 //Gana el ejercito que tiene mas soldados.
155 public static void ganador(HashMap<String,Soldado> ejercito1,HashMap<String,Soldado> ejercito2){
156     if (ejercito1.size(>ejercito2.size()){
157         System.out.println("\nGana el ejercito 1.");
158     } else if (ejercito1.size(<ejercito2.size()){
159         System.out.println("\nGana el ejercito 2.");
160     } else
161         System.out.println("\nQuedan empatados los ejercitos.");
162 }
163 }

```

II. PRUEBAS

¿Cómo comprobaste que tu práctica estaba bien?

Para verificar que todo funcionara, hice varias pruebas con diferentes datos de entrada. Por ejemplo, probé con ejércitos que tenían entre 1 y 10 soldados, y les asigné vidas que iban de 1 a 5. También revisé que las posiciones en el tablero de 10x10 fueran aleatorias y no se repitieran. Además, probé los métodos con distintos casos para asegurarme de que respondieran bien en cualquier situación.

¿Qué esperabas que pasara con cada prueba?

Esperaba que el tablero mostrara bien dónde estaban los soldados con su vida y dejara vacío donde no hubiera nadie. También que los cálculos del promedio de vida fueran correctos, dividiendo la suma de las vidas entre la cantidad de soldados de cada ejército. Quería que el soldado con más vida se identificara sin fallos, que los rankings se ordenaran de mayor a menor vida y que el método para elegir al ganador mostrara bien quién tenía más soldados o indicara empate si los ejércitos estaban igualados.

¿Qué pasó realmente con las pruebas?

Todo salió como lo esperaba. El tablero se generó correctamente con los soldados y las celdas vacías en su lugar. Los promedios de vida se calcularon bien, sin importar la cantidad de soldados. Encontré al soldado con más vida siempre que lo probé, y los rankings quedaron bien ordenados tanto con el método de burbuja como con el de inserción. Al final, el método para elegir al ganador hizo lo que debía: mostró el ejército con más soldados o señaló un empate cuando era necesario.

La primera ejecución del programa podemos ver que se generó una tabla 10x10 con 20 soldados en total. En el primer ejercito (rojo) se crearon 10 soldados donde el soldado con mayor vida es el soldado4x2 y el promedio de vida es de 3.3, mientras que en el segundo ejército (azul) se creó 10 soldados donde el soldado con mayor vida es el soldado0x7 y el promedio de vida es 3.5. Y por último nos muestra la lista de los soldados por orden de creación (por ejercito), por orden de vida (por ejercito) y el ganador de la batalla (por la cantidad de soldados por ejercito).

```

Output - Laboratorios (run)

run:

  2      4      5
      4 3      4
          5 2
          2
      5 4
  4      4      2
      1      5
          3 2
  4      3

El soldado con mayor vida del ejercito 1 es: Soldado[nombre=soldado 4X2, vida=5, fila=4, columna=2]
El soldado con mayor vida del ejercito 2 es: Soldado[nombre=soldado 0X7, vida=5, fila=0, columna=7]

El promedio de la vida del ejercito 1 es: 3.3
El promedio de la vida del ejercito 2 es: 3.5

Lista de los soldados por orden de creacion:
Ejercito 1 :
Soldado[nombre=soldado 1X3, vida=3, fila=1, columna=3]
Soldado[nombre=soldado 3X4, vida=2, fila=3, columna=4]
Soldado[nombre=soldado 4X2, vida=5, fila=4, columna=2]
Soldado[nombre=soldado 0X0, vida=2, fila=0, columna=0]
Soldado[nombre=soldado 1X2, vida=4, fila=1, columna=2]
Soldado[nombre=soldado 7X8, vida=2, fila=7, columna=8]
Soldado[nombre=soldado 5X9, vida=2, fila=5, columna=9]
Soldado[nombre=soldado 1X7, vida=4, fila=1, columna=7]
Soldado[nombre=soldado 6X5, vida=5, fila=6, columna=5]
Soldado[nombre=soldado 5X5, vida=4, fila=5, columna=5]
Ejercito 2 :
Soldado[nombre=soldado 5X0, vida=4, fila=5, columna=0]
Soldado[nombre=soldado 6X1, vida=1, fila=6, columna=1]
Soldado[nombre=soldado 0X7, vida=5, fila=0, columna=7]
Soldado[nombre=soldado 4X3, vida=4, fila=4, columna=3]
Soldado[nombre=soldado 2X4, vida=5, fila=2, columna=4]
Soldado[nombre=soldado 0X2, vida=4, fila=0, columna=2]
Soldado[nombre=soldado 9X4, vida=3, fila=9, columna=4]
Soldado[nombre=soldado 7X5, vida=3, fila=7, columna=5]
Soldado[nombre=soldado 9X0, vida=4, fila=9, columna=0]
Soldado[nombre=soldado 2X6, vida=2, fila=2, columna=6]

El ranking de los soldados es:
Ejercito 1 :
Soldado[nombre=soldado 6X5, vida=5, fila=6, columna=5]
Soldado[nombre=soldado 4X2, vida=5, fila=4, columna=2]
Soldado[nombre=soldado 5X5, vida=4, fila=5, columna=5]
Soldado[nombre=soldado 1X7, vida=4, fila=1, columna=7]
Soldado[nombre=soldado 1X2, vida=4, fila=1, columna=2]
Soldado[nombre=soldado 1X3, vida=3, fila=1, columna=3]
Soldado[nombre=soldado 5X9, vida=2, fila=5, columna=9]
Soldado[nombre=soldado 7X8, vida=2, fila=7, columna=8]
Soldado[nombre=soldado 0X0, vida=2, fila=0, columna=0]
Soldado[nombre=soldado 3X4, vida=2, fila=3, columna=4]
Ejercito 2 :
Soldado[nombre=soldado 2X4, vida=5, fila=2, columna=4]
Soldado[nombre=soldado 0X7, vida=5, fila=0, columna=7]
Soldado[nombre=soldado 9X0, vida=4, fila=9, columna=0]
Soldado[nombre=soldado 0X2, vida=4, fila=0, columna=2]
Soldado[nombre=soldado 4X3, vida=4, fila=4, columna=3]
Soldado[nombre=soldado 5X0, vida=4, fila=5, columna=0]
Soldado[nombre=soldado 7X5, vida=3, fila=7, columna=5]
Soldado[nombre=soldado 9X4, vida=3, fila=9, columna=4]
Soldado[nombre=soldado 2X6, vida=2, fila=2, columna=6]
Soldado[nombre=soldado 6X1, vida=1, fila=6, columna=1]

Quedan empatados los ejercitos.

Desea generar otros ejercitos(?) 1=Si 0=No

```

La segunda ejecución del programa podemos ver que se generó una tabla 10x10 con 9 soldados en total. En el primer ejército (rojo) se crearon 3 soldados donde el soldado con mayor vida es el soldado0x8 y el promedio de vida es de 2.3, mientras que en el segundo ejército (azul) se creó 6 soldados donde el soldado con mayor vida es el soldado6x9 y el promedio de vida es 2.83. Y por último nos muestra la lista de los soldados por orden de creación (por ejército), por orden de vida (por ejército) y el ganador de la batalla (por la cantidad de soldados por ejército).

Output - Laboratorios (run)



								5	
			3	3		2			
					2		3		
								4	
		3							
			2						

El soldado con mayor vida del ejercito 1 es: Soldado[nombre=soldado 0X8, vida=5, fila=0, columna=8]
El soldado con mayor vida del ejercito 2 es: Soldado[nombre=soldado 6X9, vida=4, fila=6, columna=9]

El promedio de la vida del ejercito 1 es: 3.3333333333333335
El promedio de la vida del ejercito 2 es: 2.8333333333333335

Lista de los soldados por orden de creacion:

Ejercito 1 :

Soldado[nombre=soldado 8X4, vida=2, fila=8, columna=4]
Soldado[nombre=soldado 7X2, vida=3, fila=7, columna=2]
Soldado[nombre=soldado 0X8, vida=5, fila=0, columna=8]

Ejercito 2 :

Soldado[nombre=soldado 3X4, vida=3, fila=3, columna=4]
Soldado[nombre=soldado 5X8, vida=3, fila=5, columna=8]
Soldado[nombre=soldado 6X9, vida=4, fila=6, columna=9]
Soldado[nombre=soldado 3X5, vida=3, fila=3, columna=5]
Soldado[nombre=soldado 5X6, vida=2, fila=5, columna=6]
Soldado[nombre=soldado 3X7, vida=2, fila=3, columna=7]

El ranking de los soldados es:

Ejercito 1 :

Soldado[nombre=soldado 0X8, vida=5, fila=0, columna=8]
Soldado[nombre=soldado 7X2, vida=3, fila=7, columna=2]
Soldado[nombre=soldado 8X4, vida=2, fila=8, columna=4]

Ejercito 2 :

Soldado[nombre=soldado 6X9, vida=4, fila=6, columna=9]
Soldado[nombre=soldado 3X5, vida=3, fila=3, columna=5]
Soldado[nombre=soldado 5X8, vida=3, fila=5, columna=8]
Soldado[nombre=soldado 3X4, vida=3, fila=3, columna=4]
Soldado[nombre=soldado 3X7, vida=2, fila=3, columna=7]
Soldado[nombre=soldado 5X6, vida=2, fila=5, columna=6]

Gana el ejercito 2.

Desea generar otros ejercitos(?) 1=Si 0=No

La tercera ejecución del programa podemos ver que se generó una tabla 10x10 con 5 soldados en total. En el primer ejército (rojo) se crearon 3 soldados donde el soldado con mayor vida es el soldado1x0 y el promedio de vida es de 4.3, mientras que en el segundo ejército (azul) se creó 2 soldados donde el soldado con mayor vida es el soldado7x8 y el promedio de vida es 3.5. Y por último nos muestra la lista de los soldados por orden de creación (por ejército), por orden de vida (por ejército) y el ganador de la batalla (por la cantidad de soldados por ejército).

Output - Laboratorios (run)



5									
								3	
							4		
		3							
									5

El soldado con mayor vida del ejercito 1 es: Soldado{nombre=soldado 1X0, vida=5, fila=1, columna=0}
 El soldado con mayor vida del ejercito 2 es: Soldado{nombre=soldado 7X8, vida=4, fila=7, columna=8}

El promedio de la vida del ejercito 1 es: 4.3333333333333333
 El promedio de la vida del ejercito 2 es: 3.5

Lista de los soldados por orden de creacion:



Ejercito 1 :
 Soldado{nombre=soldado 1X0, vida=5, fila=1, columna=0}
 Soldado{nombre=soldado 9X9, vida=5, fila=9, columna=9}
 Soldado{nombre=soldado 8X2, vida=3, fila=8, columna=2}
 Ejercito 2 :
 Soldado{nombre=soldado 7X8, vida=4, fila=7, columna=8}
 Soldado{nombre=soldado 5X9, vida=3, fila=5, columna=9}

El ranking de los soldados es:

Ejercito 1 :
 Soldado{nombre=soldado 9X9, vida=5, fila=9, columna=9}
 Soldado{nombre=soldado 1X0, vida=5, fila=1, columna=0}
 Soldado{nombre=soldado 8X2, vida=3, fila=8, columna=2}
 Ejercito 2 :
 Soldado{nombre=soldado 7X8, vida=4, fila=7, columna=8}
 Soldado{nombre=soldado 5X9, vida=3, fila=5, columna=9}

Gana el ejercito 1.

Desea generar otros ejercitos(?) 1=Si 0=No

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

III. COMMITS:

Ultimo commit

Entramos a nuestra carpeta donde están nuestros archivos y añadimos los cambios.

```

MINGW64~/c/Users/Mauro Snayder/Documents/NetBeansProjects/Laboratorios/src
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Laboratorio_08/Soldado.java
        modified:   Laboratorio_08/VideoJuego5.java

no changes added to commit (use "git add" and/or "git commit -a")

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git add .

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Laboratorio_08/Soldado.java
        modified:   Laboratorio_08/VideoJuego5.java

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```

Hacemos un commit

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git commit -m "terminado"
[master 3b612c8] terminado
 2 files changed, 26 insertions(+), 23 deletions(-)

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```

Subimos nuestro commit a nuestro repositorio remoto

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 847 bytes | 847.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/MauroSullcaMamani/FDLP2_LAB.git
   06b92e6..3b612c8  master -> master
branch 'master' set up to track 'origin/master'.

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```


Historial de todos los commits realizados para este laboratorio.

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git log
commit 3b612c839abd87637e521a9715d3592051f859a3 (HEAD -> master, origin/master)
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Fri Nov 29 00:07:22 2024 -0500

    terminado

commit 06b92e6424fdf904f6a36963675697d286f0a109
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Tue Nov 26 11:24:10 2024 -0500

    termiando la parte de imprimir los rankings

commit 7bd819ff1884a4034ef2fe3f12fb9119ca3799ac
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Tue Nov 26 11:10:10 2024 -0500

    solucionando los ordenamientos

commit 82da477d44bf78c8068b9321c418e44779de170c
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Tue Nov 26 10:07:46 2024 -0500



    avance en laboratorios
  
```

Link de mi repositorio: https://github.com/MauroSullcaMamani/FDLP2_LAB.git

IV. RUBRICA:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	✓	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	✓	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	✓	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	✓	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	✓	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
TOTAL		20		18	

Tabla 2: Rúbrica para contenido del Informe y demostración

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 10

CONCLUSIONES

En conclusión, el uso de HashMap para gestionar los ejércitos en la práctica resultó ser buena. Los HashMap permitieron organizar a los soldados de cada ejército asociándolos a claves únicas, lo que facilitó el acceso y la manipulación de datos de manera ordenada. Esta estructura de datos ofreció una forma dinámica de gestionar ejércitos con tamaños variables y permitió implementar funcionalidades clave, como calcular el promedio de vida, identificar al soldado con mayor vida y ordenar los soldados según su vida.

METODOLOGÍA DE TRABAJO

Lo primero que hice fue leer con atención el enunciado del ejercicio, asegurándome de entender las restricciones que se nos daban para poder plantear una solución adecuada al problema. Luego, analicé el código proporcionado para comprender cómo funcionaba cada parte y cómo se conectaban entre sí. Después de entenderlo, realicé las modificaciones necesarias para que cumpliera con los requerimientos del enunciado. Finalmente, comprobé que todo estuviera correcto probando varias veces para asegurarme de que el código funcionara como se esperaba en diferentes escenarios.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez and M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021, pp. 170, ISBN 978-612 5035-20-2.